Intelligent Product Builder: a rapid prototyping environment for software in context aware hardware products

Adinda Freudenthal **Marc P.A.J. de Hoogh** **David V. Keyson**
Delft University of Technology, Faculty of Industrial Design Engineering
Landbergstraat 15, Delft, The Netherlands
31 15 2783029
A.Freudenthal@io.tudelft.nl

ABSTRACT
Intelligence in products can be used to adapt user-product interaction to the context of usage. Input from sensors connected to hardware and to the user and readings from a database are processed to define the reactions of the system. However, there are no rapid prototyping tools, which support quick and easy building of software in hardware products by product-designers. Design students lack the possibility to evaluate the actual functioning of their designed intelligence with real users, since it is only expressed when tested with real hardware, in changing situations, and during longer periods of time. To support design students a rapid prototyping environment, called QuintPro Builder, is being developed at TU Delft. QuintPro Builder is currently being used in student projects for programming and to enable user tests.

KEYWORDS
Rapid software prototyping tools, interface design, context aware products, user tests

INTRODUCTION
User interfaces of intelligent products change in appearance, in interaction style and in functional properties, depending on the current context, past events and changeable data from (remote) databases. Changes in user interface presentations and interactions, have the potential to support the user in a more personalized way, to adapt to the environment or to adapt to what the user is doing. When designing products with such characteristics, we will need an analysis phase, investigating requirements and developing a design vision and concept as well as a model building and testing phase.

Specific methods for defining concepts are being developed at various universities, such as new techniques to elicit end users knowledge [e.g. 6], as well as methods to define and structure concepts and to evaluate concepts with users, e.g. by applying storyboard techniques [1].

The next step is to develop models, test with users and iterate. Building models of intelligent products requires the development of computational models. For example in the virtual Real Estate Agent (REA) a model of natural human communication was developed for allowing natural untrained input [2]. Our design students have learned to use Macromedia Director, Microsoft Visual Basic, and Maya. However, support is not provided in the form of ready-made modules towards designing intelligence in products. By programming intelligence in products we refer to building logical expressions to control product behaviour in a time and situation dependent way, managing databases (e.g. for user profiles), managing sensors coupled to hardware, the environment or the user and also building computational user models, as well as building a rich user interface experience. Programming intelligence requires trained and skilled professionals. Given the complex nature of intelligent products and the need for rapid testing and short iterative design loops, a prototyping tool, which designers and students themselves can use, is needed.

There are various partial solutions available for our purpose. Director, for instance, is primarily an animation/simulation development environment. Major programming is needed for intelligence and connecting to real hardware is difficult. Cycling 74 Max is a data flow development environment, strong in connecting sensors, but is not equipped to provide a user database and user model just like that. C++ is an industrially used language, suitable

for developing intelligent products, but programming intelligence is time consuming and changing concepts usually means major reprogramming. We, however, need an integrated solution, providing easy programming of intelligence, easy building of user profiles, easy user interface building, easy connection to hardware, while making use of the available programming skills in design students.

QuintPro Builder (QUICK INTELLIGENT PRODUCT BUILDER)

As a rapid prototyping environment for the design of intelligent products QuintPro Builder is being developed at TU Delft. QuintPro Builder provides an environment to build intelligence for the software of a hardware product. The intelligence created can function within the QuintPro Builder environment in actual prototype usage. During user tests sensors in the hardware are used as input for the software. Output in the form of events is sent to the product and to its user interface. User reactions define feedback into the user interface and from there back into the reasoning system. The user interface itself can also be defined in QuintPro Builder if it comes to simple software, i.e. programmed with VB .NET. If the designer wants more advanced interface graphics, such as dragging of interface objects and animations, a Director Shockwave movie can be embedded in the VB. NET user interface. QuintPro Builder provides the framework for data flow between sensors (connected to hardware and the user), system intelligence, database(s) and user interface(s). The user profile is stored in a database, which can be adjusted to the design by the student. The control part of the user model is one of the elements of system intelligence, and hence is designed and programmed by the design student. The *data flow connections* within the user model and QuintPro Builder provides the rest of the intelligence. Only the *intelligent behaviour* is programmed by the student.

This intelligent behaviour is based on the evaluation of conditions to determine and react to the current situation. The system compares a desirable state (of sensors or otherwise) with the actual state or history of those values. Depending on the result the system will execute commands to change the user profile, the planning data, the interface or the hardware settings. Input from the end user (from sensor or user interface) is automatically updated in the current state and history.

By applying this principle many intelligent product designs can be created. It is most suitable for designs, in which system reaction is dependent on time-based plannings, such as in a physical fitness trainer. The type of potential programmed intelligence covers a broad range: For example, in a bike fitness trainer, the heart beat frequency of the user should remain within certain boundaries during a certain time period, or else the system takes action, (e.g. a message will be sent to the user), but also selecting the most effective coaching style per individual user could be programmed by software evaluation of certain motivation level indicating values.

In Figure 1 the QuintPro Builder data flow framework and to be filled in components are shown. The components that can be programmed are indicated.

As a development environment QuintPro Builder does not only support the various components and functions. The development process of the software is supported as well. There is a building mode and a run mode. Switching between programming and testing is easy. Sensor history and event logs can be inspected for evaluation. Session recordings can be made and inspected or reused. During development connection to real sensors is not necessary. Sensor readings can be simulated manually.

The system intelligence is basically provided by conditions on the current or previous state placed in scripts that can be executed regularly. The scripts (and conditions) can be edited directly in VB. However, there is also is a dialogue box in which only the relevant variables (e.g. a sensor reading) and the logical operators have to be filled such that the script is generated automatically. There is another dialogue box available for entering commands such that execution results and logs can be inspected. (E.g. the student can define the reaction to heart beat frequency over 180 beats per minute, and then sees in the logs all executed scripts in the system). By filling this in and adding more commands the student can test the growing system step by step.

INTERACTIVE TECHNOLOGY DESIGN – A NEW MASTERS COURSE

QuintPro Builder is now in an experimental stage, and is currently being used in the new course Interactive Technology Design, taught at TU Delft. In the course the students practice developing of a clear design vision, towards designing human centred functionality by means of user-product-social interaction. They learn to integrate knowledge on new technologies (e.g. sensors, system learning) into design opportunities, and to translate this knowledge into design parameters and they learn to build prototypes of experiential quality and to test these with users, iteratively.

In this years course we decided to let the students design an intelligent bike trainer. Students were given the choice of designing a bike interface and coach for revalidation or for sports purposes. An actual bike trainer, equipped with sensors to measure heart beat frequency, pedal rotation

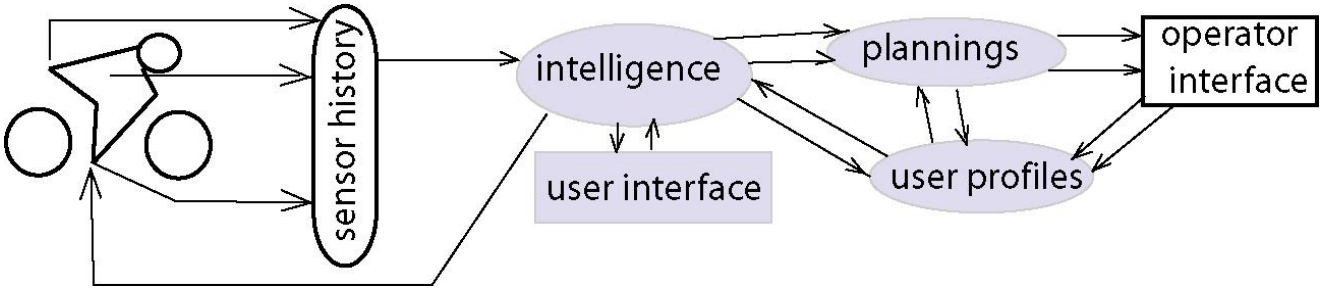

Figure 1 This figure illustrates the relations between the physical bike trainer, the intelligence created and user interface created, as it is used during user testing. The user interface can either be programmed by using QuintPro Builder or in Macromedia Director or by a combination of the two. The bike and user are equipped with sensors. The QuintPro Builder data flow framework and coupled designed system, manage the data flow and command execution. Input can be given either by an operator or the bike user (even/also simultaneously). The third user interface to QuintPro Builder, that for the programming designer, allows changing of the four grey coloured components. The set of sensors is not fixed. If the designer wants another type of product or interaction, other sensors can be used.

frequency, and usage of the seat is connected to a computer. The pedal resistance level can be set by the computer. The computer is connected to a touch screen on the handlebar which presents the user interface. The students' task is to design the multimodal user interface (sound and graphics) and interaction principles and dialogues.

For the assignment we have chosen a product, which commercially is available with a fairly sophisticated level of intelligence, especially if user data are stored on a central computer and users can be identified by a personal ID token. In some gyms there are already personalized, adaptive systems in use. Monitoring training levels through heart beat sensors is commonplace. In the design assignment we asked the students to design intelligence beyond the current solutions. They were asked to develop prototypical scenarios of interaction, representative for their design view, and relating to certain intelligent reasoning. The scenarios were to be programmed and tested by the students with users. The students were not asked to program a complete bike intelligence and interface, but rather to develop and test intelligence and interaction principles.

INTERIM FINDINGS AND DIRECTIONS FOR THE FUTURE

We are now halfway through the course. Students have been working with an alpha version of the program. This is a tricky situation. We have already found some interesting consequences for design education. Because the students know that they will have to implement their ideas into a working prototype their approach seems to become more focused. For the first time we see that students are becoming aware of what intelligence means and what the consequence for interaction design is.

At the same time thinking about product reactions in terms of computational reasoning is very new for them, and consequently we have already changed our lectures to provide much more detailed and explicit information about designing intelligence they seem to need now. We have seen that students need practice in transforming design visions into computational models.

The QuintPro environment and its tutorial are being tested intensely via usage by the design students. We have already received excellent feedback for improvements. At the SIGCHI.nl 2004 conference we hope to present several example designs and students' experiences on QuintPro Builder as an educational tool and as a concept development tool.

Once QuintPro Builder has been further developed we can begin to expand our past research on intelligence in products. In the past we have, for example, developed an intelligent thermostat interface [4] and a medical device tutor [5]. Both supposedly would adapt to the user's level of device operation experience. Also other intelligent behaviour has been the topic of investigations, such as observation of the location of inhabitants in their homes, and resulting system suggestions in a dialogue to adjust their thermostat settings.

To support designers in defining and testing their designed context sensitive dialog, as in the thermostat, we have developed and tested a rapid prototyping tool [3]. This tool supports the assessment of a current state in a hierarchically structured set of procedural tasks and actions to define the appropriate system dialogue. QuintPro Builder, in contrast, mainly aims at supporting the linking of dynamic data flow from sensors (from hardware/the environment/the user) and manages the real time system and interface reactions.

There is a gap between industrial prototype testing, with customer panels and the available level of simulated

functionality and interaction in considering proposals for advanced intelligent products. The basic concept ideas cannot be tested. Intelligence must be tested in the field, over a longer period of time and with a range of users. However, since it is unclear as to what properties of intelligence are best choices in terms optimal functionality and user acceptance, as well as marketing values, such high-risk developments are not engaged into easily. This is probably one of the reasons why we still see so few products in the market with the advanced type of intelligence we are looking for.

With QuintPro Builder we hope to develop a rapid prototyping environment for designers and researchers, which allows for quick building and evaluating in real interaction with various users, real hardware, in different situations, and during longer periods of time. This will give us the possibility to test more intelligence and interface principles with a much higher iteration speed. This will stimulate the process of generating hypothesis and testing as well as building up design experience.

ACKNOWLEDGMENTS

We thank Marion de Groot for her support in developing QuintPro Builder and our students for being patient and helpful.

REFERENCES

1. Carroll, J.M. (2000) Making use: Scenarios and scenario-based design. In: Interfacing Reality in the New Milennium, OZCHI 2000, Conference Proceedings, 4th-8th December 2000, Sydney, Australia, pp. 36-48.
2. Cassell, J. (2000) Nudge Nudge Wink Wink: Elements of Face-to-Face Conversation for Embodied Conversational Agents. In: Cassel, J., Sullivan, J., Prevost, S., Churchill, E., Embodied conversational agents, Massachusetts Institute of Technology, pp. 1-63, 2000.
3. DeKoven, E, de Hoogh, M. and Keyson, D. (2003). Rapid prototyping collaborative dialogue interfaces. Human-Computer Interaction - INTERACT '03. M. Rauterberg et al (Eds.). IOS Press.
4. Freudenthal, A., Keyson, D.V., Dekoven, E., de Hoogh, M.P.A.J. (2001) Communicating extensive smart home functionality to users of all ages: the design of a mixed-initiative multimodal thermostat interface. In: OIKOS 2001 Workshop: Methodological Issues in the Design of Household Technologies, Molslaboratoriet, Denmark, March 12-13, 34-39.
5. Freudenthal, A., van der Weide-Mook, H.J., Bouwman, C.A.H.M., Glasbeek, H.A., Snijders, C.J. (2004) Leren werken met een medisch apparaat: Het meester-gezel principe toegepast in een interactieve product simulatie, In: De Anesthesiemedewerker: volwassen en zelfstandig!? 21e Nationaal Anesthesiesymposium, Utrecht, the Netherlands, 17 Januari 2004.
6. Sanders, E.B.N. (2000) Generative Tools for CoDesigning. In: Collaborative Design, Scrivener, Ball and Woodcock (Eds.) (London: Springer-Verlag), 3-12